Final: Pendulum Project

Qijian Liu

April 9, 2021

**Introduction:**

In the experiment, I built a pendulum and recorded its motions in various different setups in videos. The videos were process through a computer program, named Tracker, and the data of the pendulum’s angle versus time is obtained. Finally, the data were curve fitted using the theoretical model and compared. It is found that the model only fits the data for a small segment in the beginning, and over predicts the effects of damping for the remainder. After refitting the data, a new equation is found that better predicts the motion, although still not perfectly. Also, the equation for period is not exactly correct, and a new equation was also introduced to amend it.

**Theory:**

The pendulum built undergoes damped harmonic oscillation when it oscillates. It is predicted that the motion of the pendulum is predicted to obey the following equation from the manual:

Where is the initial angle, *t* is the time, *T* is the period, is the decay constant, and is the phase constant; values of angles are in radians.

It is also predicted that the period will obey the following equation from the manual:

Where *L* and *D* are the length of the string and the distance to center of mass from the point of attachment, respectively.

**Setup Description:**

The pendulum was built by tying a string to a rod. The string was selected and tied in a way such that in the course of the oscillation, the length of the string will not change (see figure 1). The mass was chosen to be a stack of disks; one stack will consist of 5 disks (see figure 2). The string wrapped around the disks through the hole in the middle when attached. When it is needed to increase the mass, add another stack to the original. This way, the center of mass remains the same. When it is needed to adjust the length, wrap the string around the disks more to shorten it, less to lengthen it. A protractor is attached to the rod and perpendicular to the string so that the initial angle can be measured. Finally, the rod is elevated enough so that the pendulum can swing. Another rod is placed directly below the pendulum for calibration of the program. See figure 3 for full setup.



Figure 1: String tied to rod.



Figure 2: Disks stacks example



Figure 3: Full setup sample.

**Methods:**

Two stacks of disks are prepared. The experiment was done in 2 different lengths, 3 different angles, and 2 different masses for a total of 12 different setups. The camera was placed parallel to the plane that the pendulum swings in. Recording started before the start of each oscillations and edited so that it begins with the releasing of the mass, so .

The 3 initial angles are and . The 2 different lengths (*L* + *D*) are 68.5 cm and 47 cm.

For each setup, the mass was released carefully so that it swings in a plane as much as possible. The motion was recorded for at least 30 seconds in each setup. Alternate the starting points from left to right between each setup.

After data were collected from all the setups, the videos were processed through the Tracker program. After calibration, it automatically tracks the motion of the pendulum and gives the angle with respect to time. These data were curve fitted in Python to see how well the model given fits the actual data.

**Results:**

The following are the plots of the angle (in radians) versus time for each of the setups, as well as the curve fit function:

|  |  |
| --- | --- |
| Trial 1: Length: 0.685m  Angle:  Mass:  1 stack |  |
| Trial 2: Length: 0.685m  Angle:  Mass:  1 stack |  |
| Trial 3: Length: 0.685m  Angle:  Mass:  1 stack |  |
| Trial 4: Length: 0.685m  Angle:  Mass:  2 stack |  |
| Trial 5: Length: 0.685m  Angle:  Mass:  2 stack |  |
| Trial 6: Length: 0.685m  Angle:  Mass:  2 stack |  |
| Trial 7: Length: 0.470m  Angle:  Mass:  1 stack |  |
| Trial 8: Length: 0.470m  Angle:  Mass:  1 stack |  |
| Trial 9: Length: 0.470m  Angle:  Mass:  2 stack |  |
| Trial 10: Length: 0.470m  Angle:  Mass:  2 stack |  |
| Trial 11: Length: 0.470m  Angle:  Mass:  2 stack |  |
| Trial 12: Length: 0.470m  Angle:  Mass:  2 stack |  |

For unknown reasons, Python was not able to correctly determine the decay constant for plot 2, 3, 4, 9, and 10, as seen from the plots that their fits are extremely poor compared to others. These curve fits will be omitted.

Decay constants and reduced chi-squared for the remainder of the plots are found to be:

|  |  |  |
| --- | --- | --- |
| Plot | Decay constant | Reduced Chi-Squared |
| 1 | 9.361 | 64.1 |
| 5 | 9.639 | 224.0 |
| 6 | 4.539 | 52.9 |
| 7 | 3.430 | 82.4 |
| 8 | 3.794 | 158.6 |
| 11 | 7.427 | 299.0 |
| 12 | 8.884 | 44.4 |

From the plots, the periods of oscillations can be found:

|  |  |  |
| --- | --- | --- |
| Plot | Average period from plot (s) | Period from equation (s) |
| 1 | 1.684 | 1.655 |
| 2 | 1.696 |
| 3 | 1.696 |
| 4 | 1.686 |
| 5 | 1.689 |
| 6 | 1.680 |
| 7 | 1.395 | 1.371 |
| 8 | 1.394 |
| 9 | 1.390 |
| 10 | 1.390 |
| 11 | 1.397 |
| 12 | 1.394 |

**Analysis**:

While the periods from the equation are closed to the average periods found from the plot, the uncertainty range does not include any of the average periods found. It is unknown what the uncertainty on the time difference between each frame for the camera is, but it should not be large enough such that it lies within the uncertainty of the equation. On average, the predicted values from the equation are about 0.03s less than the average periods. If the equation is changed to:

The predicted periods become s and s, respectively. After changing the factor, the the equation and the average periods of the plots agree.

From the periods, it is also evident that changing the mass has no effect on the period. Additionally, it is also evident that the initial angle does not affect the period. Because the phase constant is just when the start of the recording is, it also does not affect the period. These are just as predicted from the equation.

The pendulum is symmetric. Since from the plots, the motions are similar, whether the pendulum starts on the left or on the right.

However, it is clear from the plots and the reduced chi-squared that the theoretical model does not fit the data very well. Namely, the amplitude decays much quicker than the actual motion. Therefore, curve fit is performed on the data again, this time against the following equation:

The following are the results of the refitting:

|  |  |
| --- | --- |
| Reduced Chi-squared | Plots |
| 9.8 |  |
| 116.4 |  |
| 26.9 |  |
| 70.5 |  |
| 5.5 |  |
| 9.3 |  |
| 24.7 |  |
| 38.9 |  |
| N/A |  |
| N/A |  |
| 8.0 |  |
| 3.4 |  |

Once again, Python is unable to predict the parameters for plots 9 and 10. Their results will be excluded.

These are still not very good fit, as their reduced chi-squared indicate. However, they are significantly better than the original model. Furthermore, the shape of the curve fits the data, other than plots 2 and 3.

The parameter *b* varies greatly in all the fits. Therefore, this should be a variable. The parameter *a* ranges from 0.976 to 1.00, with a huge uncertainty for all of them. The parameter *c* is averaged to be 3.69 for the first 6 plots, and 4.48 for the remaining plots. If they are multiplied by their respective periods (1.648 for the first one, 1.371 for the second one), the results are 6.2308 and 6.1421 respectively. The average becomes 6.1864, slightly less than the original constant of .

Finally, using the parameters from the refitting, a better model of the pendulum is:

This indicates that the pendulum amplitude does not decay exponentially.

The main source of error that causes the fit to be less accurate, even in the new model, could be the tracking program’s imperfection. Sometimes, the program does not track the mass properly; instead, it moves slightly off from the intended tracking target. It can be seen on some of the plots that the oscillates oddly. For example, in plot 3, after about 4 seconds, the whole oscillation shifted upward as a result of the program’s tracking issue. With better equipment producing better data, the fit should be more accurate.

**Conclusion:**

From the experiment, it can be concluded that the model provided does not predict the pendulum that was built well. It over predicts how the amplitude decays because the pendulum does not actually decay exponentially. Instead, the base is about 0.988 with parameter *b* positive. Additionally, it can also be concluded that the predicted period is slightly less than the actual period. Finally, the experiment confirms that the initial angles, mass, and the phase constant has no effect on the period.

**Appendix:**

Link to track program used: <https://physlets.org/tracker/>

Program 1:

# Curve fitting the provided model

import numpy as np

import matplotlib.pyplot as plt

from scipy.optimize import curve\_fit

t1, x1, y1, pheta1, err1 = np.loadtxt('1 Data.csv', delimiter=',', skiprows = 2, unpack = True)

t2, x2, y2, pheta2, err2 = np.loadtxt('2 Data.csv', delimiter=',', skiprows = 2, unpack = True)

t3, x3, y3, pheta3, err3 = np.loadtxt('3 Data.csv', delimiter=',', skiprows = 2, unpack = True)

t4, x4, y4, pheta4, err4 = np.loadtxt('4 Data.csv', delimiter=',', skiprows = 2, unpack = True)

t5, x5, y5, pheta5, err5 = np.loadtxt('5 Data.csv', delimiter=',', skiprows = 2, unpack = True)

t6, x6, y6, pheta6, err6 = np.loadtxt('6 Data.csv', delimiter=',', skiprows = 2, unpack = True)

t7, x7, y7, pheta7, err7 = np.loadtxt('7 Data.csv', delimiter=',', skiprows = 2, unpack = True)

t8, x8, y8, pheta8, err8 = np.loadtxt('8 Data.csv', delimiter=',', skiprows = 2, unpack = True)

t9, x9, y9, pheta9, err9 = np.loadtxt('9 Data.csv', delimiter=',', skiprows = 2, unpack = True)

t10, x10, y10, pheta10, err10 = np.loadtxt('10 Data.csv', delimiter=',', skiprows = 2, unpack = True)

t11, x11, y11, pheta11, err11 = np.loadtxt('11 Data.csv', delimiter=',', skiprows = 2, unpack = True)

t12, x12, y12, pheta12, err12 = np.loadtxt('12 Data.csv', delimiter=',', skiprows = 2, unpack = True)

trial, com, pheta, mass = np.loadtxt('Data.csv', delimiter=',', skiprows = 1, unpack = True)

def convert\_rad (deg):

return deg\*np.pi/180

def T(com):

return 2\*(com)\*\*(1/2)

def chi\_square(y\_list, f\_output, sigma):

chi\_sqr = 0

for i in range (0, len(y\_list)):

chi\_sqr += ((y\_list[i] - f\_output[i])/sigma[i])\*\*2

return chi\_sqr

pheta1 = convert\_rad(pheta1)

pheta2 = convert\_rad(pheta2)

pheta3 = convert\_rad(pheta3)

pheta4 = convert\_rad(pheta4)

pheta5 = convert\_rad(pheta5)

pheta6 = convert\_rad(pheta6)

pheta7 = convert\_rad(pheta7)

pheta8 = convert\_rad(pheta8)

pheta9 = convert\_rad(pheta9)

pheta10 = convert\_rad(pheta10)

pheta11 = convert\_rad(pheta11)

pheta12 = convert\_rad(pheta12)

com = com/100

pheta = convert\_rad(pheta)

period = T(com)

def pendulum1 (t, tau):

return pheta1[0]\*np.exp(-t/tau)\*np.cos(2\*np.pi\*(t/period[0]))

def pendulum2 (t, tau):

return pheta2[0]\*np.exp(-t/tau)\*np.cos(2\*np.pi\*(t/period[1]))

def pendulum3 (t, tau):

return pheta3[0]\*np.exp(-t/tau)\*np.cos(2\*np.pi\*(t/period[2]))

def pendulum4 (t, tau):

return pheta4[0]\*np.exp(-t/tau)\*np.cos(2\*np.pi\*(t/period[3]))

def pendulum5 (t, tau):

return pheta5[0]\*np.exp(-t/tau)\*np.cos(2\*np.pi\*(t/period[4]))

def pendulum6 (t, tau):

return pheta6[0]\*np.exp(-t/tau)\*np.cos(2\*np.pi\*(t/period[5]))

def pendulum7 (t, tau):

return pheta7[0]\*np.exp(-t/tau)\*np.cos(2\*np.pi\*(t/period[6]))

def pendulum8 (t, tau):

return pheta8[0]\*np.exp(-t/tau)\*np.cos(2\*np.pi\*(t/period[7]))

def pendulum9 (t, tau):

return pheta9[0]\*np.exp(-t/tau)\*np.cos(2\*np.pi\*(t/period[8]))

def pendulum10 (t, tau):

return pheta10[0]\*np.exp(-t/tau)\*np.cos(2\*np.pi\*(t/period[9]))

def pendulum11 (t, tau):

return pheta11[0]\*np.exp(-t/tau)\*np.cos(2\*np.pi\*(t/period[10]))

def pendulum12 (t, tau):

return pheta12[0]\*np.exp(-t/tau)\*np.cos(2\*np.pi\*(t/period[11]))

############################

popt , pcov = curve\_fit(pendulum1, t1, pheta1, (10), convert\_rad(err1), True)

pvar = np.diag(pcov)

plt.figure(figsize=(10,10))

plt.scatter(t1, pheta1)

plt.plot(t1, pendulum1(t1, \*popt), color='red', marker='|')

plt.xlabel("t (s)")

plt.ylabel("pheta (rad)")

plt.title('Plot 1')

print(popt, ' +- ', np.sqrt(pvar))

y\_list = np.zeros(len(pheta1))

for i in range (0, len(pheta1)):

y\_list[i] = pendulum1(t1[i], \*popt)

v = len(t1)-len(popt)

chi = chi\_square(y\_list, pheta1, convert\_rad(err1))

print(chi/v)

popt , pcov = curve\_fit(pendulum2, t2, pheta2, (10), convert\_rad(err2), True)

pvar = np.diag(pcov)

plt.figure(figsize=(10,10))

plt.scatter(t2, pheta2)

plt.plot(t2, pendulum2(t2, \*popt), color='red', marker='|')

plt.xlabel("t (s)")

plt.ylabel("pheta (rad)")

plt.title('Plot 2')

print(popt, ' +- ', np.sqrt(pvar))

y\_list = np.zeros(len(pheta2))

for i in range (0, len(pheta2)):

y\_list[i] = pendulum2(t2[i], \*popt)

v = len(t2)-len(popt)

chi = chi\_square(y\_list, pheta2, convert\_rad(err2))

print(chi/v)

popt , pcov = curve\_fit(pendulum3, t3, pheta3, (10), convert\_rad(err3), True)

pvar = np.diag(pcov)

plt.figure(figsize=(10,10))

plt.scatter(t3, pheta3)

plt.plot(t3, pendulum3(t3, \*popt), color='red', marker='|')

plt.xlabel("t (s)")

plt.ylabel("pheta (rad)")

plt.title('Plot 3')

print(popt, ' +- ', np.sqrt(pvar))

y\_list = np.zeros(len(pheta3))

for i in range (0, len(pheta3)):

y\_list[i] = pendulum3(t3[i], \*popt)

v = len(t3)-len(popt)

chi = chi\_square(y\_list, pheta3, convert\_rad(err3))

print(chi/v)

popt , pcov = curve\_fit(pendulum4, t4, pheta4, (10), convert\_rad(err4), True)

pvar = np.diag(pcov)

plt.figure(figsize=(10,10))

plt.scatter(t4, pheta4)

plt.plot(t4, pendulum4(t4, \*popt), color='red', marker='|')

plt.xlabel("t (s)")

plt.ylabel("pheta (rad)")

plt.title('Plot 4')

print(popt, ' +- ', np.sqrt(pvar))

y\_list = np.zeros(len(pheta4))

for i in range (0, len(pheta4)):

y\_list[i] = pendulum4(t4[i], \*popt)

v = len(t4)-len(popt)

chi = chi\_square(y\_list, pheta4, convert\_rad(err4))

print(chi/v)

popt , pcov = curve\_fit(pendulum5, t5, pheta5, (10), convert\_rad(err5), True)

pvar = np.diag(pcov)

plt.figure(figsize=(10,10))

plt.scatter(t5, pheta5)

plt.plot(t5, pendulum5(t5, \*popt), color='red', marker='|')

plt.xlabel("t (s)")

plt.ylabel("pheta (rad)")

plt.title('Plot 5')

print(popt, ' +- ', np.sqrt(pvar))

y\_list = np.zeros(len(pheta5))

for i in range (0, len(pheta5)):

y\_list[i] = pendulum5(t5[i], \*popt)

v = len(t5)-len(popt)

chi = chi\_square(y\_list, pheta5, convert\_rad(err5))

print(chi/v)

popt , pcov = curve\_fit(pendulum6, t6, pheta6, (10), convert\_rad(err6), True)

pvar = np.diag(pcov)

plt.figure(figsize=(10,10))

plt.scatter(t6, pheta6)

plt.plot(t6, pendulum6(t6, \*popt), color='red', marker='|')

plt.xlabel("t (s)")

plt.ylabel("pheta (rad)")

plt.title('Plot 6')

print(popt, ' +- ', np.sqrt(pvar))

y\_list = np.zeros(len(pheta6))

for i in range (0, len(pheta6)):

y\_list[i] = pendulum6(t6[i], \*popt)

v = len(t6)-len(popt)

chi = chi\_square(y\_list, pheta6, convert\_rad(err6))

print(chi/v)

popt , pcov = curve\_fit(pendulum7, t7, pheta7, (10), convert\_rad(err7), True)

pvar = np.diag(pcov)

plt.figure(figsize=(10,10))

plt.scatter(t7, pheta7)

plt.plot(t7, pendulum7(t7, \*popt), color='red', marker='|')

plt.xlabel("t (s)")

plt.ylabel("pheta (rad)")

plt.title('Plot 7')

print(popt, ' +- ', np.sqrt(pvar))

y\_list = np.zeros(len(pheta7))

for i in range (0, len(pheta7)):

y\_list[i] = pendulum7(t7[i], \*popt)

v = len(t7)-len(popt)

chi = chi\_square(y\_list, pheta7, convert\_rad(err7))

print(chi/v)

popt , pcov = curve\_fit(pendulum8, t8, pheta8, (10), convert\_rad(err8), True)

pvar = np.diag(pcov)

plt.figure(figsize=(10,10))

plt.scatter(t8, pheta8)

plt.plot(t8, pendulum8(t8, \*popt), color='red', marker='|')

plt.xlabel("t (s)")

plt.ylabel("pheta (rad)")

plt.title('Plot 8')

print(popt, ' +- ', np.sqrt(pvar))

y\_list = np.zeros(len(pheta8))

for i in range (0, len(pheta8)):

y\_list[i] = pendulum8(t8[i], \*popt)

v = len(t8)-len(popt)

chi = chi\_square(y\_list, pheta8, convert\_rad(err8))

print(chi/v)

popt , pcov = curve\_fit(pendulum9, t9, pheta9, (10), convert\_rad(err9), True)

pvar = np.diag(pcov)

plt.figure(figsize=(10,10))

plt.scatter(t9, pheta9)

plt.plot(t9, pendulum9(t9, \*popt), color='red', marker='|')

plt.xlabel("t (s)")

plt.ylabel("pheta (rad)")

plt.title('Plot 9')

print(popt, ' +- ', np.sqrt(pvar))

y\_list = np.zeros(len(pheta9))

for i in range (0, len(pheta9)):

y\_list[i] = pendulum9(t9[i], \*popt)

v = len(t9)-len(popt)

chi = chi\_square(y\_list, pheta9, convert\_rad(err9))

print(chi/v)

popt , pcov = curve\_fit(pendulum10, t10, pheta10, (10), convert\_rad(err10), True)

pvar = np.diag(pcov)

plt.figure(figsize=(10,10))

plt.scatter(t10, pheta10)

plt.plot(t10, pendulum10(t10, \*popt), color='red', marker='|')

plt.xlabel("t (s)")

plt.ylabel("pheta (rad)")

plt.title('Plot 10')

print(popt, ' +- ', np.sqrt(pvar))

y\_list = np.zeros(len(pheta10))

for i in range (0, len(pheta10)):

y\_list[i] = pendulum10(t10[i], \*popt)

v = len(t10)-len(popt)

chi = chi\_square(y\_list, pheta10, convert\_rad(err10))

print(chi/v)

popt , pcov = curve\_fit(pendulum11, t11, pheta11, (10), convert\_rad(err11), True)

pvar = np.diag(pcov)

plt.figure(figsize=(10,10))

plt.scatter(t11, pheta11)

plt.plot(t11, pendulum11(t11, \*popt), color='red', marker='|')

plt.xlabel("t (s)")

plt.ylabel("pheta (rad)")

plt.title('Plot 11')

print(popt, ' +- ', np.sqrt(pvar))

y\_list = np.zeros(len(pheta11))

for i in range (0, len(pheta11)):

y\_list[i] = pendulum11(t11[i], \*popt)

v = len(t11)-len(popt)

chi = chi\_square(y\_list, pheta11, convert\_rad(err11))

print(chi/v)

popt , pcov = curve\_fit(pendulum12, t12, pheta12, (10), convert\_rad(err12), True)

pvar = np.diag(pcov)

plt.figure(figsize=(10,10))

plt.scatter(t12, pheta12)

plt.plot(t12, pendulum12(t12, \*popt), color='red', marker='|')

plt.xlabel("t (s)")

plt.ylabel("pheta (rad)")

plt.title('Plot 12')

print(popt, ' +- ', np.sqrt(pvar))

y\_list = np.zeros(len(pheta12))

for i in range (0, len(pheta12)):

y\_list[i] = pendulum12(t12[i], \*popt)

v = len(t12)-len(popt)

chi = chi\_square(y\_list, pheta12, convert\_rad(err12))

print(chi/v)

'''

Output:

[9.36287364] +- [0.0791974]

64.08940909094852

[0.5367109] +- [0.01234515]

169.0434475573074

[0.45698091] +- [0.03791648]

40.47299959672531

[0.46046883] +- [0.01944343]

148.43271288731768

[9.63909736] +- [0.05718269]

224.05093943780216

[4.53886781] +- [0.09859894]

52.94208461145377

[3.43017211] +- [0.05402449]

82.40464881696677

[3.79424235] +- [0.03518235]

158.63642004451353

[0.08201131] +- [inf]

47.932268683117506

[0.17314689] +- [0.01227037]

123.49344119269654

[7.42671617] +- [0.04600884]

299.04959398653756

[8.8836067] +- [0.1219697]

44.37953428656363

'''

Program 2:

# Refitting for new model

import numpy as np

import matplotlib.pyplot as plt

from scipy.optimize import curve\_fit

t1, x1, y1, pheta1, err1 = np.loadtxt('1 Data.csv', delimiter=',', skiprows = 2, unpack = True)

t2, x2, y2, pheta2, err2 = np.loadtxt('2 Data.csv', delimiter=',', skiprows = 2, unpack = True)

t3, x3, y3, pheta3, err3 = np.loadtxt('3 Data.csv', delimiter=',', skiprows = 2, unpack = True)

t4, x4, y4, pheta4, err4 = np.loadtxt('4 Data.csv', delimiter=',', skiprows = 2, unpack = True)

t5, x5, y5, pheta5, err5 = np.loadtxt('5 Data.csv', delimiter=',', skiprows = 2, unpack = True)

t6, x6, y6, pheta6, err6 = np.loadtxt('6 Data.csv', delimiter=',', skiprows = 2, unpack = True)

t7, x7, y7, pheta7, err7 = np.loadtxt('7 Data.csv', delimiter=',', skiprows = 2, unpack = True)

t8, x8, y8, pheta8, err8 = np.loadtxt('8 Data.csv', delimiter=',', skiprows = 2, unpack = True)

t9, x9, y9, pheta9, err9 = np.loadtxt('9 Data.csv', delimiter=',', skiprows = 2, unpack = True)

t10, x10, y10, pheta10, err10 = np.loadtxt('10 Data.csv', delimiter=',', skiprows = 2, unpack = True)

t11, x11, y11, pheta11, err11 = np.loadtxt('11 Data.csv', delimiter=',', skiprows = 2, unpack = True)

t12, x12, y12, pheta12, err12 = np.loadtxt('12 Data.csv', delimiter=',', skiprows = 2, unpack = True)

trial, com, pheta, mass = np.loadtxt('Data.csv', delimiter=',', skiprows = 1, unpack = True)

def convert\_rad (deg):

return deg\*np.pi/180

def T(com):

return 2\*(com)\*\*(1/2)

def chi\_square(y\_list, f\_output, sigma):

chi\_sqr = 0

for i in range (0, len(y\_list)):

chi\_sqr += ((y\_list[i] - f\_output[i])/sigma[i])\*\*2

return chi\_sqr

pheta1 = convert\_rad(pheta1)

pheta2 = convert\_rad(pheta2)

pheta3 = convert\_rad(pheta3)

pheta4 = convert\_rad(pheta4)

pheta5 = convert\_rad(pheta5)

pheta6 = convert\_rad(pheta6)

pheta7 = convert\_rad(pheta7)

pheta8 = convert\_rad(pheta8)

pheta9 = convert\_rad(pheta9)

pheta10 = convert\_rad(pheta10)

pheta11 = convert\_rad(pheta11)

pheta12 = convert\_rad(pheta12)

com = com/100

pheta = convert\_rad(pheta)

period = T(com)

def pendulum1 (t, a, b, c):

return pheta1[0]\*a\*\*(b\*t)\*np.cos(c\*t)

def pendulum2 (t, a, b, c):

return pheta2[0]\*a\*\*(b\*t)\*np.cos(c\*t)

def pendulum3 (t, a, b, c):

return pheta3[0]\*a\*\*(b\*t)\*np.cos(c\*t)

def pendulum4 (t, a, b, c):

return pheta4[0]\*a\*\*(b\*t)\*np.cos(c\*t)

def pendulum5 (t, a, b, c):

return pheta5[0]\*a\*\*(b\*t)\*np.cos(c\*t)

def pendulum6 (t, a, b, c):

return pheta6[0]\*a\*\*(b\*t)\*np.cos(c\*t)

def pendulum7 (t, a, b, c):

return pheta7[0]\*a\*\*(b\*t)\*np.cos(c\*t)

def pendulum8 (t, a, b, c):

return pheta8[0]\*a\*\*(b\*t)\*np.cos(c\*t)

def pendulum9 (t, a, b, c):

return pheta9[0]\*a\*\*(b\*t)\*np.cos(c\*t)

def pendulum10 (t, a, b, c):

return pheta10[0]\*a\*\*(b\*t)\*np.cos(c\*t)

def pendulum11 (t, a, b, c):

return pheta11[0]\*a\*\*(b\*t)\*np.cos(c\*t)

def pendulum12 (t, a, b, c):

return pheta12[0]\*a\*\*(b\*t)\*np.cos(c\*t)

############################

popt , pcov = curve\_fit(pendulum1, t1, pheta1, (1, 1, 1), convert\_rad(err1), True)

pvar = np.diag(pcov)

plt.figure(figsize=(10,10))

plt.scatter(t1, pheta1)

plt.plot(t1, pendulum1(t1, \*popt), color='red', marker='|')

plt.xlabel("t (s)")

plt.ylabel("pheta (rad)")

plt.title('Plot 1')

print(popt, ' +- ', np.sqrt(pvar))

y\_list = np.zeros(len(pheta1))

for i in range (0, len(pheta1)):

y\_list[i] = pendulum1(t1[i], \*popt)

v = len(t1)-len(popt)

chi = chi\_square(y\_list, pheta1, convert\_rad(err1))

print(chi/v)

popt , pcov = curve\_fit(pendulum2, t2, pheta2, (1, 1, 1), convert\_rad(err2), True)

pvar = np.diag(pcov)

plt.figure(figsize=(10,10))

plt.scatter(t2, pheta2)

plt.plot(t2, pendulum2(t2, \*popt), color='red', marker='|')

plt.xlabel("t (s)")

plt.ylabel("pheta (rad)")

plt.title('Plot 2')

print(popt, ' +- ', np.sqrt(pvar))

y\_list = np.zeros(len(pheta2))

for i in range (0, len(pheta2)):

y\_list[i] = pendulum2(t2[i], \*popt)

v = len(t2)-len(popt)

chi = chi\_square(y\_list, pheta2, convert\_rad(err2))

print(chi/v)

popt , pcov = curve\_fit(pendulum3, t3, pheta3, (1, 1, 1), convert\_rad(err3), True)

pvar = np.diag(pcov)

plt.figure(figsize=(10,10))

plt.scatter(t3, pheta3)

plt.plot(t3, pendulum3(t3, \*popt), color='red', marker='|')

plt.xlabel("t (s)")

plt.ylabel("pheta (rad)")

plt.title('Plot 3')

print(popt, ' +- ', np.sqrt(pvar))

y\_list = np.zeros(len(pheta3))

for i in range (0, len(pheta3)):

y\_list[i] = pendulum3(t3[i], \*popt)

v = len(t3)-len(popt)

chi = chi\_square(y\_list, pheta3, convert\_rad(err3))

print(chi/v)

popt , pcov = curve\_fit(pendulum4, t4, pheta4, (1, 1, 1), convert\_rad(err4), True)

pvar = np.diag(pcov)

plt.figure(figsize=(10,10))

plt.scatter(t4, pheta4)

plt.plot(t4, pendulum4(t4, \*popt), color='red', marker='|')

plt.xlabel("t (s)")

plt.ylabel("pheta (rad)")

plt.title('Plot 4')

print(popt, ' +- ', np.sqrt(pvar))

y\_list = np.zeros(len(pheta4))

for i in range (0, len(pheta4)):

y\_list[i] = pendulum4(t4[i], \*popt)

v = len(t4)-len(popt)

chi = chi\_square(y\_list, pheta4, convert\_rad(err4))

print(chi/v)

popt , pcov = curve\_fit(pendulum5, t5, pheta5, (1, 1, 1), convert\_rad(err5), True)

pvar = np.diag(pcov)

plt.figure(figsize=(10,10))

plt.scatter(t5, pheta5)

plt.plot(t5, pendulum5(t5, \*popt), color='red', marker='|')

plt.xlabel("t (s)")

plt.ylabel("pheta (rad)")

plt.title('Plot 5')

print(popt, ' +- ', np.sqrt(pvar))

y\_list = np.zeros(len(pheta5))

for i in range (0, len(pheta5)):

y\_list[i] = pendulum5(t5[i], \*popt)

v = len(t5)-len(popt)

chi = chi\_square(y\_list, pheta5, convert\_rad(err5))

print(chi/v)

popt , pcov = curve\_fit(pendulum6, t6, pheta6, (1, 1, 1), convert\_rad(err6), True)

pvar = np.diag(pcov)

plt.figure(figsize=(10,10))

plt.scatter(t6, pheta6)

plt.plot(t6, pendulum6(t6, \*popt), color='red', marker='|')

plt.xlabel("t (s)")

plt.ylabel("pheta (rad)")

plt.title('Plot 6')

print(popt, ' +- ', np.sqrt(pvar))

y\_list = np.zeros(len(pheta6))

for i in range (0, len(pheta6)):

y\_list[i] = pendulum6(t6[i], \*popt)

v = len(t6)-len(popt)

chi = chi\_square(y\_list, pheta6, convert\_rad(err6))

print(chi/v)

popt , pcov = curve\_fit(pendulum7, t7, pheta7, (1, 1, 1), convert\_rad(err7), True)

pvar = np.diag(pcov)

plt.figure(figsize=(10,10))

plt.scatter(t7, pheta7)

plt.plot(t7, pendulum7(t7, \*popt), color='red', marker='|')

plt.xlabel("t (s)")

plt.ylabel("pheta (rad)")

plt.title('Plot 7')

print(popt, ' +- ', np.sqrt(pvar))

y\_list = np.zeros(len(pheta7))

for i in range (0, len(pheta7)):

y\_list[i] = pendulum7(t7[i], \*popt)

v = len(t7)-len(popt)

chi = chi\_square(y\_list, pheta7, convert\_rad(err7))

print(chi/v)

popt , pcov = curve\_fit(pendulum8, t8, pheta8, (1, 1, 1), convert\_rad(err8), True)

pvar = np.diag(pcov)

plt.figure(figsize=(10,10))

plt.scatter(t8, pheta8)

plt.plot(t8, pendulum8(t8, \*popt), color='red', marker='|')

plt.xlabel("t (s)")

plt.ylabel("pheta (rad)")

plt.title('Plot 8')

print(popt, ' +- ', np.sqrt(pvar))

y\_list = np.zeros(len(pheta8))

for i in range (0, len(pheta8)):

y\_list[i] = pendulum8(t8[i], \*popt)

v = len(t8)-len(popt)

chi = chi\_square(y\_list, pheta8, convert\_rad(err8))

print(chi/v)

popt , pcov = curve\_fit(pendulum9, t9, pheta9, (1, 1, 1), convert\_rad(err9), True)

pvar = np.diag(pcov)

plt.figure(figsize=(10,10))

plt.scatter(t9, pheta9)

plt.plot(t9, pendulum9(t9, \*popt), color='red', marker='|')

plt.xlabel("t (s)")

plt.ylabel("pheta (rad)")

plt.title('Plot 9')

print(popt, ' +- ', np.sqrt(pvar))

y\_list = np.zeros(len(pheta9))

for i in range (0, len(pheta9)):

y\_list[i] = pendulum9(t9[i], \*popt)

v = len(t9)-len(popt)

chi = chi\_square(y\_list, pheta9, convert\_rad(err9))

print(chi/v)

popt , pcov = curve\_fit(pendulum10, t10, pheta10, (1, 1, 1), convert\_rad(err10), True)

pvar = np.diag(pcov)

plt.figure(figsize=(10,10))

plt.scatter(t10, pheta10)

plt.plot(t10, pendulum10(t10, \*popt), color='red', marker='|')

plt.xlabel("t (s)")

plt.ylabel("pheta (rad)")

plt.title('Plot 10')

print(popt, ' +- ', np.sqrt(pvar))

y\_list = np.zeros(len(pheta10))

for i in range (0, len(pheta10)):

y\_list[i] = pendulum10(t10[i], \*popt)

v = len(t10)-len(popt)

chi = chi\_square(y\_list, pheta10, convert\_rad(err10))

print(chi/v)

popt , pcov = curve\_fit(pendulum11, t11, pheta11, (1, 1, 1), convert\_rad(err11), True)

pvar = np.diag(pcov)

plt.figure(figsize=(10,10))

plt.scatter(t11, pheta11)

plt.plot(t11, pendulum11(t11, \*popt), color='red', marker='|')

plt.xlabel("t (s)")

plt.ylabel("pheta (rad)")

plt.title('Plot 11')

print(popt, ' +- ', np.sqrt(pvar))

y\_list = np.zeros(len(pheta11))

for i in range (0, len(pheta11)):

y\_list[i] = pendulum11(t11[i], \*popt)

v = len(t11)-len(popt)

chi = chi\_square(y\_list, pheta11, convert\_rad(err11))

print(chi/v)

popt , pcov = curve\_fit(pendulum12, t12, pheta12, (1, 1, 1), convert\_rad(err12), True)

pvar = np.diag(pcov)

plt.figure(figsize=(10,10))

plt.scatter(t12, pheta12)

plt.plot(t12, pendulum12(t12, \*popt), color='red', marker='|')

plt.xlabel("t (s)")

plt.ylabel("pheta (rad)")

plt.title('Plot 12')

print(popt, ' +- ', np.sqrt(pvar))

y\_list = np.zeros(len(pheta12))

for i in range (0, len(pheta12)):

y\_list[i] = pendulum12(t12[i], \*popt)

v = len(t12)-len(popt)

chi = chi\_square(y\_list, pheta12, convert\_rad(err12))

print(chi/v)

'''

Output:

[0.9939172 3.79070342 3.7267891 ] +- [1.54322660e+02 9.59218387e+04 1.60271531e-04]

9.765348140915622

[0.97655654 2.17157968 3.6575878 ] +- [5.55165932e+02 5.20401518e+04 2.08482055e-04]

116.40212523845567

[0.99472977 3.91410776 3.68293051] +- [3.14759488e+02 2.32659901e+05 3.23682766e-04]

26.900996000348908

[0.99765831 6.40848977 3.66440535] +- [8.44424937e+01 2.27311039e+05 1.75779992e-04]

70.48735515026064

[ 1.00556032 -1.71133591 3.71624646] +- [6.87864965e+02 2.11844390e+05 1.07563806e-04]

5.53263345170657

[0.99814093 3.15583583 3.69697403] +- [4.63430817e+02 7.87423391e+05 2.51341061e-04]

9.281622881826383

[0.99541332 5.24558127 4.47138021] +- [1.22475130e+02 1.39811010e+05 2.08966908e-04]

24.743661588107233

[0.99592747 8.3692736 4.46747535] +- [4.63222027e+01 9.50805587e+04 1.81887880e-04]

38.88146079131104

[0.97799913 9.86676659 0.83750292] +- [3.87301868e+03 1.69266526e+06 1.00391778e-02]

47.97639771121943

[ 0.98040014 11.21074696 0.59903031] +- [7.90478288e+02 4.43702011e+05 2.69679443e-03]

117.99912948148916

[0.9977323 3.21200511 4.48952988] +- [9.01720641e+01 1.27866339e+05 6.42402243e-05]

7.963336629943003

[ 0.99998079 486.39883226 4.50536657] +- [1.05200982e-02 2.65107063e+05 1.66651805e-04]

3.370298495931352

'''